BOLT BERANEK AND NEWMAN INC

CONSULTING • DEVELOPMENT • RESEARCH

Report No. 2396

July 1972

INTERFACE MESSAGE PROCESSORS FOR THE ARPA COMPUTER NETWORK

QUARTERLY TECHNICAL REPORT NO. 14
1 April 1972 to 30 June 1972

QTRLY:#13-AD740709

Principal Investigator: Mr. Frank E. Heart
Telephone (617) 491-1850, Ext. 470

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 1260

Contract No. DAHC15-69-C-0179
Effective Date: 2 January 1969
Expiration Date: 31 December 1972
Contract Amount: $7,256,300

Title of Work: IMP

Submitted to:

Director
Advanced Research Projects Agency
Arlington, Virginia 22209

23

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Mass. 02138

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE Interface Message Processors For The ARPA Computer Network

QUARTERLY TECHNICAL REPORT NO. 14

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

Bolt Beranek and Newman Inc.

6. REPORT DATE
July 1972

7a. TOTAL NO. OF PAGES
18

7b. NO. OF REFS

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

BBN Report No. 2396

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Advanced Research Projects Agency
Arlington, Virginia 22209

13. ABSTRACT

The basic function of the ARPA computer network is to allow large existing computers (Hosts), with different system configurations, to communicate with each other. Each Host is connected to an Interface Message Processor (IMP), which transmits messages from its Host(s) to other Hosts and accepts messages for its Host(s) from other Hosts. There is frequently no direct communication circuit between two Hosts that wish to communicate; in these cases intermediate IMPs act as message switchers. The message switching is performed as a store and forward operation. The IMPs regularly exchange information which: allows each IMP to adapt its message routing to the conditions of its local section of the network; reports network performance and malfunctions to a Network Control Center; permits message tracing so that network operation can be studied comprehensively; allows network reconfiguration without reprogramming each IMP. The Terminal IMP (TIP), which consists of an IMP and a Multi-Line Controller (MLC), extends the network concepts by permitting the direct attachment (without an intervening Host) of up to 64 dissimilar terminal devices to the network. The Terminal IMP program provides many aspects of the Host protocols in order to allow effective communication between a terminal user and a Host process.

ia

UNCLASSIFIED
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computers and Communication						
Store and Forward Communication						
ARPA Computer Network						
Interface Message Processor						
IMP						
Terminal IMP						
TIP						
Honeywell DDP-516						
Honeywell DDP-316						
Multi-Line Controller						
MLC						
Network Control Center						
Host Protocol						
High Speed Modular IMP						
HSMIMP						
Lockheed SUE						

ik

INTERFACE MESSAGE PROCESSORS FOR THE ARPA COMPUTER NETWORK

QUARTERLY TECHNICAL REPORT NO. 14

1 April 1972 to 30 June 1972

Submitted to:

Advanced Research Projects Agency
Arlington, Virginia 22209
Attn: Dr. L.G. Roberts

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract No. DAHC-15-69-C-0179.

ic

TABLE OF CONTENTS

1. OVERVIEW

This Quarterly Technical Report, Number 14, describes aspects of our work on the ARPA Computer Network during the second quarter of 1972.

During this quarter there have been a large number of equipment deliveries. A 316 IMP was installed at Ft. Belvoir (Virginia) and four TIPs were installed at: Global Weather Central (GWC), Omaha, Nebraska; National Oceanographic and Atmospheric Agency (NOAA), Boulder, Colorado; Seismic Array Analysis Center (SAAC), Alexandria, Virginia; and ARPA, Arlington, Virginia. The TIP installed at GWC included the magnetic tape option. In addition, BBN fabricated and installed two special Host interfaces, one for a PDP-10 at AMES and one for a PDP-15 at ARPA. A fourth Host interface was added to the IMP at MIT, which is thus the first IMP in the network to be attached to four server Hosts. We have also purchased a number of TIP terminals and installed them at ETAC, SAAC, and ARPA.

During the second quarter we made four attempts to release the new software system described in Section 4 of our Quarterly Technical Report Number 13. Although, for a variety of reasons, this system is not yet operating in the field, we believe that the software will be operational early in the third quarter. We have learned a great deal about the procedures required to effect major network software revisions from our release attempts; some of these operational problems and procedures are described in Section 2.

Another major activity during the third quarter was the continuing HSMIMP design effort. By the end of the quarter we had completed an evaluation of a large number of contenders for selection as the HSMIMP processor, selected the Lockheed SUE, and submitted a purchase order for sufficient processors and other equipment to construct a full-scale prototype machine. We also made major progress toward the design of the memory access switch required by the HSMIMP configuration. Details of the switch design, the processor selection procedure, and the SUE processor are provided in Section 3.

We intensified our investigation of the requirements on the "Satellite IMP (SIMP)" during the second quarter. It is currently anticipated that several IMPs will communicate with each other via an earth-satellite relay over a single broadcast channel (or pair of channels). Thus, the SIMP's must not only provide additional buffering as required by a long-delay communication circuit (see Section 3 of our Quarterly Technical Report Number 13) but also provide special algorithms to effect the efficient sharing of the total channel capacity. We have been actively investigating various algorithms in close cooperation with representatives of the University of Hawaii's ALOHA system and with the ARPA staff.

By the end of the second quarter we had expanded our NCC staff to provide three-shift coverage seven days a week. This increased coverage has already proven useful in reducing total IMP down time during the last few weeks of June. We anticipate that this expanded staffing will increase in value as the network grows across time-zone boundaries.

During the second quarter we published an updated version of BBN Report No. 1822, *Specifications for the Interconnection of a Host and an IMP*. This update dealt primarily with the "Very Distant Host" interface (see our Quarterly Technical Report Number 12). We also submitted a paper for presentation at the 1972 Fall Joint Computer Conference; this paper describes the new algorithms for transmission and flow control in the network (see our Quarterly Technical Report Number 13) as well as the evolution of the IMP software structure.

Finally, we have continued to respond to user requests for additions and modifications to the TIP software. During the second quarter a number of new TIP commands were added, including the functions of:

- performing "binary" I/O at a TIP terminal
- setting up a device to always accept "requests for connection" from anywhere in the network
- providing (optionally) a "linefeed" after a user types "carriage-return"
- accepting commands to the TIP from the Host end of a connection, rather than only from the terminal end.

We also began implementation of a revised set of commands for controlling local/remote character echoing which seem to be closer to the desires of TIP users. In addition, we are participating in a study of the issues involved in "remote-controlled" echoing, as a possible TIP option.

2. NCC OPERATION

2.1 Release of New System Software

For several quarters, the many changes made to the running version of IMP software (IMPSYS) were relatively minor, and the incompatibilities between old and new versions at each such change were relatively trivial. The last time a *major* change was made to IMPSYS the network consisted of less than 15 machines, all of which were the same model. By the end of this quarter the network consisted of 29 machines which included:

- DDP-516's and H-316's
- IMP's and TIP's
- TIP's with and without special features (e.g., magnetic tape option or tailored buffer layout).

During the second quarter we attempted, on four occasions, to make a major change to the running IMPSYS. None of these attempts was completely successful, but each helped us refine our procedures for making such changes in a complex operating environment "on the fly."

In our initial release attempt we grossly underestimated the difficulty of coordinating operators scattered across the country.

- Each running IMPSYS contains a section of code (IMPLOD) which can, if activated, reload the IMP from a neighbor.

The versions of IMPLOD for the old and new systems at the time of the first attempt were incompatible. Accordingly, well in advance of the scheduled release we mailed a paper tape copy of the new IMPLOD to each site, and later confirmed by telephone that it had been received.

We planned to reload the Net by telephoning each site and asking that the new copy of IMPLOD be manually loaded and started, at each site in sequence. It proved impossible to reach a person at each site (in the right order) as promptly as we wished. One site could not even find the tape, although we had previously confirmed that the correct person had received it.

- Both the old and the new systems "poisoned" each other. This means that neither of two adjacent IMP's would run successfully if they were running opposite versions of IMPSYS. Our solution to the "poisoning" problem was to stop an IMP while the adjacent IMP was reloaded with the new version, then reloading this one only after the IMP(s) on the other side had been halted. As with the IMPLOD tape loading, this was a procedure which had to be coordinated by telephone, thus increasing the difficulty associated with the need to contact individuals at each site at the appropriate time.

Our conclusions from this effort, implemented effectively with the next attempt, were as follows.

Conclusion 1:

Site operator assistance must not be required at every site during the release of a new software system. This implies that versions of IMPLOD *must* be kept compatible and that releases of new IMPLOD tapes can therefore be asynchronous with regard to new system software releases. With regard to our release procedure, we abandoned the recently mailed IMPLOD tape. We modified the new version of IMPSYS so that its IMPLOD section was close enough (although not identical) to the old version; as the new IMPLOD section overlaid the old, the system *kept running* even though a second reload (based on parameters in the reloaded IMPLOD section) might be required. The result of this new policy is that changes to IMPLOD can now be made for reasons of convenience to the BBN operators, increased foolproofing of the reloading process, etc., but need not be closely synchronized with changes in the IMPSYS itself.

Conclusion 2:

There must be a simple direct way to control whether a new and an old IMPSYS "talk" to each other when they are adjacent in the network. In this case, both versions were modified not to do so; in other cases the opposite may be appropriate. In the leader of each message in the new system, we set aside the bit corresponding to the old system's "Trace" bit and always set this bit on. The Trace facility in the old system was then inactivated at the time of each release attempt, so that this bit would unambiguously define the source of each message as an "old" system or a "new" system. (The Trace facility is used only by the Network Measurement Center, and the change was coordinated with the NMC in every case.)

For changes of version, such as this current one, where the two versions do not intercommunicate, the procedure for reloading the IMP's is very complex. It consists of reloading IMP's one-by-one, commanded from IMP #5 at the NCC in every case where possible. The procedure is monitored by keeping two IMP's #5 on the network for the duration of the change, one running the "old" IMPSYS and the other running the "new" version (and connected to the NCC machine). From the "old" IMP #5 we command each IMP to reload itself from a neighbor which is running the new system, and that IMP reports itself as reloaded and up to the NCC through the "new" IMPSYS subset of the network.

In case of line or IMP failures which coincide with the release, the technique is extended to apply to IMP's (or sets of IMP's) which have only one path to BBN as follows. First, each IMP in the set is patched so as *not* to reload any neighbor. Next, each is ordered to request a reload from its neighbor closest to the "new" system. Eventually an entire arm of the network is "dead", with each IMP trying to reload. Finally, the IMP with two or more paths to BBN is reloaded from an adjacent IMP running the new IMPSYS. This IMP then reloads his neighbor (his "don't reload anybody" patch having been overlaid by the reload), the neighbor subsequently reloads his neighbor(s), etc., until all IMPs in the arm have been reloaded.

Our second attempt to release the new software followed the procedures described above and was successful in distributing the new software. However, operation of the system for several hours revealed malfunctions which had not been observed during checkout at BBN. The new system was left operational until the malfunctions were diagnosed. The old system was then returned to operation successfully, again by following the procedures described above.

The third and fourth release attempts followed the same pattern, but involving the TIP (terminal handling) rather than IMP (store-and-forward) software. During each release, it is necessary to first disable the TIP portion of the software in each TIP, then propagate the new IMPSYS, then the new TIP software, and finally re-enable the TIP code. The stage of releasing the TIP software was never reached during the first two release attempts. By the third attempt, we discovered that evolution of the new IMPSYS and the TIP software had diverged to such an extent that it was impossible to release and enable the TIP code. This was primarily due to the fact that the IMPSYS and the TIP code did not communicate through a clean interface; rather, the memory locations, use of interrupts, etc., which formed the interface grew in an ad hoc way as features were added to the TIP code. This led to the third operating principle.

Conclusion 3:

Even though IMPSYS and TIP software are tightly coupled and co-resident in a single machine, the interface at all levels must be carefully controlled. In practice, this has meant removing much of the use of interrupts from the TIP code, as well as centralizing the memory locations used for communication between IMPSYS and the TIP.

After changes to the structure of the TIP software, a fourth release attempt was made. Both IMP's and TIP's appeared to function satisfactorily for several hours, but after a day of operation we observed that TIP's were failing at the rate of about one failure in the network during an hour (corresponding to failures in a single TIP of about once every 8-10 hours). Again, the software was left operational until enough data was gathered to diagnose the causes of failure, and the old system was then revived.

Based on the steadily decreasing number of problems with each release attempt, we are quite hopeful of a successful release of the new system early in the third quarter.

2.2 Other NCC Changes

During the second quarter we have increased operator coverage to 24-hours per day, every day. We have also increased telephone lines for NCC contact to two, with additional lines reserved for later expansion. We as yet have found no economic solution to the problem of easy toll-free inward calling from all sites; INWATS does not seem economic in this case.

Our new Host (an old but very reliable PDP-1 time-sharing machine) has taken over the task of reloading TIP's. The result is faster and more flexible than previous procedures based on paper tape loading, and does not interfere with the development usage of TIP 30. We are in the process of implementing a general NCP for support of the various control and monitoring functions we have in mind for the PDP-1 Host, although there are no plans to provide a general server facility on it.

3. HIGH SPEED MODULAR IMP

During the first half of 1972 we have been considering the design of the High Speed Modular IMP (HSMIMP), with emphasis on the impact of the choice of vendor on our original design ideas. During this period our concept of the structure of the HSMIMP has changed somewhat, particularly in the area of the "switch", from that presented in our Quarterly Technical Report No. 12. We have considered a number of vendors as potential suppliers of processors, memories, and switch modules; vendors considered included, among others, Data General, DEC, Honeywell, Lockheed, and Varian.

3.1 Design Concepts

The design goal of the prototype HSMIMP is to achieve a throughput increase of a factor of ten over the DDP-516 IMP with a 1 μsec cycle. We believe that the most sensible approach to this goal is a multiprocessor configuration of mini computers and I/O multiplexors, each with access to shared memory through a high-speed switch. In order to compute how many processors of a given type would be necessary to achieve this power, a crude benchmark was developed. The results of this benchmark were scaled by a factor of merit derived from the programming niceties of the machine, such as multiple accumulators, index registers, large page size, etc. This factor ranged from 2 to 1/2 for the machines considered, taking the 516 as 1. From this benchmark, the number of processors needed to be the equivalent of ten 516s was computed. This number ranged from 3 to 250; of the processors which seem to merit serious investigation, the range was 6 to 17.

All of these values are based on the assumption that a processor's memory references take the same length of time as they would out of that processor's normal memory. This time varies from 300 ns to about 10 μsec; the processors seriously considered were clustered around 850 ns. If the delay for a memory reference — through the switch, queueing delays, memory delay, and back through the switch — exceeds this time, each processor will be slowed and thus more processors will be required.

The critical nature of the switch-delay time can be relieved by adding some local memory to each processor; each processor could then execute code from, and maintain private temporary storage in, its own local memory. Given this configuration, consideration of the IMP program shows that most memory references can be to the local memory. The number of external references is down by a factor of 4 or more from the number of total references; thus bandwidth requirements for the external memories are down by a factor of 4, as is the effect of switch delay. Therefore, a switch delay of 200 ns with an 800 ns memory will slow the system by only 6% rather than 25%.

An implementation which requires each processor to execute code out of a local, rather than a global, memory does place some constraints on the program, but these are not as severe as might be guessed. The objective of having any processor able to do any task (thus giving reliability in the case of single or multiple processor failures, as well as versatility in being able to shift the computational power to bear on the instantaneous problem) can be realized in this scheme as easily as in the no-local-memory scheme by keeping a copy of the time critical code in each of the local memories. Infrequently used code will be loaded from shared memory to a local memory as needed. The size requirement of the local memory is not excessive; the IMP's time-critical code can fit in 4K words with room to spare.

The concept of code-sharing out of mass memory has also been proposed as an argument against local memory. This argument is not particularly telling, however, since code sharing would probably not be practical in any case. The bandwidths available from the most cost-effective memories today are roughly comparable to the speeds of the most cost-effective processors. This would generally require that there be as many independent copies of the "hot" code as there are processors, each resident in a separate memory module.

3.2 Switch Design

Various schemes have been considered for the design of the Memory Access Switch. Our original idea was to attempt to construct a single (expandable) crossbar-type switch. We now feel that the scheme which is simplest conceptually, probably the simplest to design, and perhaps the fastest in operation is a complete set of "bus couplers". Each bus coupler consists of a master (or processor-like) device and a slave (or memory-like) device connected through a cable. The slave device is connected to a processor's bus (i.e., it is a slave to the processor) and the master device is connected to the bus of a memory module (i.e., the memory is a slave to it).

When a processor makes a reference to a shared memory module, the slave end of the appropriate bus coupler attached to that processor's bus is activated (by recognizing the address) and makes a request through the cable. The master end of the bus coupler then makes a request on the memory bus to which it is connected. When the memory becomes available, the data transfer takes place through the bus coupler and the processor's bus is released.

The Memory Access Switch may be built from a set of these couplers connecting each of the processors, including the IOP(s), to each of the shared memories. If the Interface control registers are made to appear as locations in the memory address space, this same bus coupler can be used to connect the program processors to the I/O Processor(s). This scheme is convenient for machines (e.g., the Lockheed SUE and the DEC PDP-11) which normally communicate with their I/O devices in this fashion; it can be made to work on other machines of more conventional architecture (e.g., the Data General NOVA).

The disadvantages of a switch such as this are the cost and number of cables. For a system with P program processors, M external memories, and I I/O processors, this scheme requires $P \cdot M + P \cdot I + I \cdot M$ Bus Couplers. A complete "Black Box" type switch would require only $P + M + I$ cables. Other schemes, such as a 4-in 2-out switch module, require an intermediate number of cables. For very large values of P, M, and I, the associated savings are great. A prototype HSMIMP might have 6 program processors, 4 memories, and 1 IOP, for a total of 34 bus couplers. If these can be fabricated for $500 each, the total switch cost is then $17,000. This seems reasonable, considering:

1) Such a switch is ultimately modular, and can be expanded with ease or contracted to a very small minimum;

2) Only two types of modules (1 slave, 1 master) need be designed;

3) The prototype HSMIMP is probably large compared to most IMPs that might be built, the switch cost for these smaller IMPs would be substantially lower;

4) Differert switch schemes for much larger values of M, P, and I, may be tested or implemented, using multiple levels of Bus Couplers combined with Bus Arbiters for switching at the nodes.

5) We are investigating the possibility of designing a bus coupler consisting of one slave device and an indefinite number of master devices. This scheme is electrically more complex but mechanically simpler (it is conceptually identical) and represents a potential saving in cost and overall complexity. The results of this investigation, however, do not affect the basic HSMIMP design.

The HSMIMP thus appears as shown in Figure 1, with enough program processors to give the required computational power, enough I/O processors to handle the required I/O bandwidth, and enough Memory modules to handle the processors roughly every fourth cycle, plus the I/O.

3.3 Processor Selection Criteria

A number of computers have been considered for the position of HSMIMP program processor. They were compared on a number of criteria, some quantitative, some qualitative. Some of the areas of comparison were:

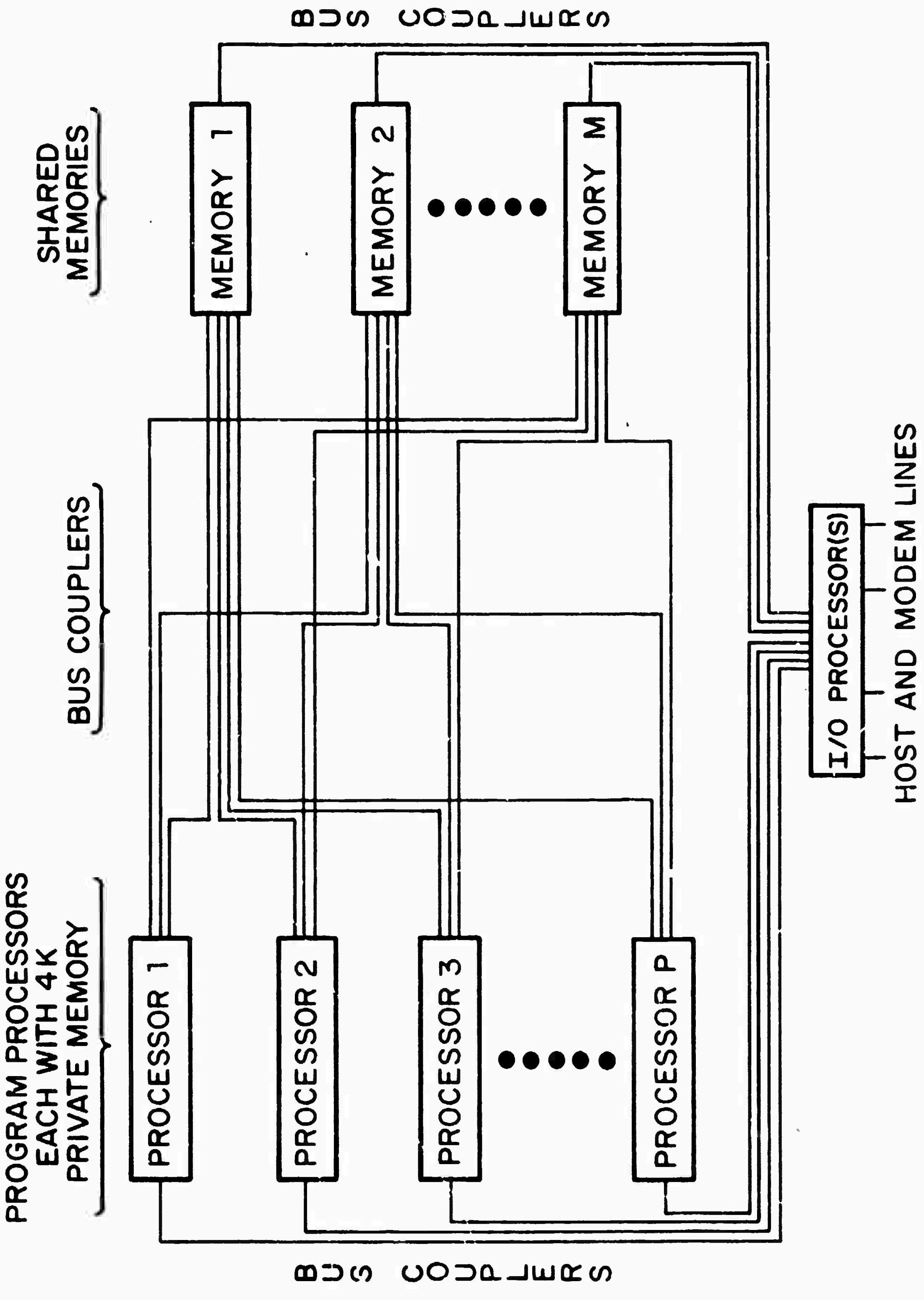

FIG. 1 HSMIMP CONFIGURATION

A) How much does a system cost, and how big is it physically, for systems of various powers?

B) How hard is it to design and build a prototype system?

C) How hard is it to produce and maintain one? How much down time can we expect?

Within these areas, a number of specific criteria were considered. In category A, we considered the cost and size of the prototype HSMIMP (equivalent to ten 516's), and the cost and size of a minimal configuration. Both costs and sizes were for a system: the prototype HSMIMP estimates included 32K words of shared memory, a basket and power supply for the IOP, and as many Bus Couplers as necessary; the minimal system estimates included 16K words of memory, a Teletype with interface, and enough spare rcom for IMP special interfaces.

In category B, we considered the logical, electrical, and "knowability" characteristics of the memory bus, as a measure of the difficulty of designing the Bus Coupler; the number of such processors in existence, as a measure of how much debugging of the processor we might have to do; and the number of special modules we would have to design.

In category C, we considered again the number of such processors in existence, since low probability processor bugs tend to decrease in number as more processors are in the field; the believability of the machine design, more easily measured for those processors of which a large number have been in the field for a long time; the corporate believability, since small companies have a greater propensity towards disappearance and towards undebugged machines; and the distance from Cambridge, since it is easier to get machines to Cambridge by simple truck than by air freight, etc., and also easier to ship them back in case of trouble.

Application of these criteria to a large number of processors, both existing and announced, led us to select the Lockheed SUE processor for the construction of the HSMIMP.

3.4 The Lockheed SUE Processor

The Lockheed SUE is a new, modern, slow (4 microsecond add), microprogrammed, inexpensive (4K version $3975) modular computer. It is unusually modular in that the various components plug in to a high-speed bus, called the INFIBUS, which is the base of the system, and the bus arbitration is done by a *separate* module, independent of the processor. A processor bids for a bus cycle much like any other device; thus multiple processors may run on a given bus. The high-speed nature of the bus is such as to permit this multi-processing with minimal interference.

The INFIBUS (like the PDP-11 UNIBUS) is homogeneous: no distinction is made between memory and I/O buses or locations. The separate bus arbiter relieves us of the need to design an arbitration scheme for the memory banks in addition to one for the I/O Processor. The high-speed bus has another attractive attribute, in that it permits the use of multiple external memories on a single bus, thus substantially decreasing the number of Bus Couplers needed. A single-processor SUE can provide a low cost minimal IMP configuration, while a multiplicity of dual-processor (i.e., two processors per bus) SUE units yields a low HSMIMP cost and a competitive size, at less than 84 inches of rack space, despite the requirement for 12-14 processors. This large number of processors diminishes the amount of bandwidth lost by a single processor failure.

The system design is attractive, taking advantage of the homogeneous high speed bus structure, with a separate arbiter. The public nature of this bus predicts that our interconnection scheme will not need to be changed because of a change in bus protocol. Expansion to larger systems appears straightforward, due to the asynchronous operation, the attractive electrical characteristics of the bus, and the ability to place multiple processors on a bus. The program can be made uniform between large and small systems, since no distinction is made between memory locations and I/O device registers.